**EDA215 Digital- och datorteknik Z**

**EDA311 Digital- och datorteknik E**

**Gamla och nya kursen**

# Tentamen

Tisdag 10 april 2007, kl. 08.30 - 12.30 i V-salar

---

**Examinatorer**
Rolf Snedsböl, tel. 772 1665

**Kontaktpersoner under tentamen**
Som ovan.

**Tillåtna hjälpmedel**

Häftet

*Instruktionslista för FLEX*

och

*Instruktionslista för MC6809*

eller

*Instruktionslista för CPU12*

I dessa får rättelser och understrykningar vara införda, inget annat.

Tabellverk och miniräknare får ej användas!

**Allmänt**
Siffror inom parentes anger full poäng på uppgiften. **Full poäng kan fås om**:

- redovisningen av svar och lösningar är läslig och tydlig. **OBS!** Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift.
-

  din lösning ej är onödigt komplicerad.
- du motiverat dina val och ställningstaganden
- redovisningen av en hårdvarukonstruktion innehåller funktionsbeskrivning, lösning och realisering.
- redovisningen av en mjukvarukonstruktion i assembler är fullständigt dokumenterad, d v s är redovisad både i strukturform (flödesplan eller pseudospråk) och med kommenterat program i assemblerspråk, om inget annat anges i uppgiften.

**Betygsättning**
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända. Tentamen ger slutbetyget:

20p ≤ **betyg 3** < 30p ≤ **betyg 4** < 40p ≤ **betyg 5**

**Lösningar**
anslås på kursens www hemsida.

**Betygslistan**
anslås såsom anges på kursens hemsida.

**Granskning**
Tid och plats anges på kursens hemsida.



**Avdelningen för datorteknik**
**Institutionen data- och informationsteknik**
**CHALMERS TEKNISKA HÖGSKOLA**

## 1. Talomvandling, aritmetik och flaggor

I uppgift a-d nedan används 5-bitars tal. X=00111 och Y=10000

**a.** Visa med penna och papper hur räkneoperationen R = X-Y utförs i en dator (i en ALU). **(1p)**

**b.** Ange sedan flaggbitarna N, Z, V, C **(1p)**

**c.** Tolka bitmönstren R, X och Ysom tal *utan* tecken och ange dess decimala motsvarighet **(1p)**

**d.** Tolka bitmönstren R, X och Y som tal *med* tecken och ange dess decimala motsvarighet. **(1p)**

**e.** Det hexadecimala talet 36 svarar mot ett 6-bitars tvåkomplementstal.
Ange dess decimala motsvarighet. **(1p)**

**f.** Omvandla det decimala talet 113 till motsvarande NBCD-tal. **(1p)**

**g.** Ett 7-bitars tvåkomplementstal (11011111) är utökad med en paritetsbit.
Ange om udda eller jämn paritet används och motivera svaret. **(1p)**

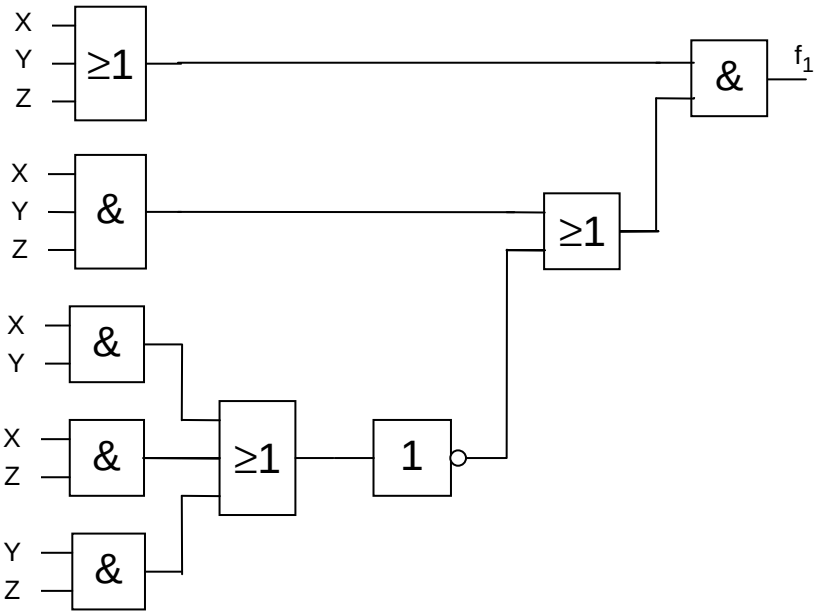
## 2. Digitalteknik, kombinatoriska nät

**a.** Man behöver en 2-ingångs NAND-grind men har tillgång till enbart 2- ingångs OR-grindar (inte ens inverterare finns). Hur realiserar du ”NAND-grinden” enbart med OR? **(2p)**

**b.** Undersök i tabellform med hjälp av fullständig binär evaluering om utrycken f(xyz)=g(xyz), där f=xy+y'z och g=y'+x'z **(2p)**

**c.** Ange funktionstabellen och rita symbolen för en 1 av 4 väljare. **(2p)**

**d.** Konstruktion. Studera kopplingen i figur 1. Om det går, minimera denna koppling och rita det nya nätet. Du har endast tillgång till NOR-grindar med valfritt antal ingångar. **(6p)**




Figur 1

**3. Digitalteknik, sekvensnät**

**a.** Visa hur du kan bilda en JK-vippa med hjälp av en SR-vippa och två AND-grindar. **(2p)**

**b.** Konstruktion. En räknare ger utsekvensen **0, 2, 4, 6**, 0, 2, osv. på utsignalerna $Q_2$, $Q_1$, $Q_0$.

Rita en tillståndsgraf för räknaren. Realisera räknaren.
Du har tillgång till INVERTERARE, NAND grindar och T-vippor med positiv flanktriggning. Du kan bortse från hur räknaren skall startas. **(6p)**

---

**4. Styrenheten för FLEX**

**a.** I tabellen nedan visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLEX-processorns instruktioner. NF i tabellens sista rad anger att nästa tillstånd (state) skall vara det första i FETCH-sekvensen. Rita en tabell där du anger State nr (0..4) och Styrsignaler. Endast styrsignaler = 1 skall anges.
Du kan utelämna RTN-beskrivningen i din tabell **(1p)**

| State nr | RTN-beskrivning | Styrsignaler (=1) |
|---|---|---|
| 0 | PC→MA, PC+1→PC | |
| 1 | M→MA | |
| 2 | M→T | |
| 3 | A+T →R, Flaggor→CC | |
| 4 | R→A, NF | |

**b.** Förklara i ord vad instruktionen ovan utför i varje klockcykel.
Skriv instruktionen med assemblerspråk. **(2p)**

**c.** Rita en tabell motsvarande den ovan, som visar utförandefasen för maskininstruktionen **JSR ,X** för FLEX-processorn. I instruktionslistan för FLEX-processorn beskrivs instruktionen enligt följande tabell. **(5p)**

| **Instruktion** | | **Adressering** | | | **Operations-beskrivning*** | **Flaggor** | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Operation** | **Beteckning** | **Absolute** | | | | **3** | **2** | **1** | **0** |
| | | **OP** | **#** | **~** | | **N** | **Z** | **V** | **C** |
| Jump to subroutine | JSR ,X | 9A | 1 | 6 | S–1 → S<br>PC → M(S)<br>X → PC | • | • | • | • |

---

**5. Småfrågor och assemblerprogrammering för FLEX**

**a.** Styrenheten för en av FLEX-datorerna i labbet är trasig. Alla DEC-instruktioner (DECA, DECB, DEC Adr, etc) fungerar inte som tänkt. Samma sak gäller för alla SUB-instruktioner, ingen fungerar som tänkt. Däremot fungerar alla övriga instruktioner korrekt. Kan du ange en instruktionssekvens som utför DEC Adr?

När instruktionssekvensen är klar skall registerinnehållen ha samma värden som innan instruktionssekvensen påbörjats (Självklart får CC och PC ändras). **(3p)**

**b.** FLEX-instruktionen BEQ LOOP (2 bytes, och OP-kod = $5D) har sin OP-kod på adress $C9 i minnet. LOOP har adressen $AD. Ange maskininstruktionen för BEQ-instruktionen. Visa hur du resonerat. **(3p)**

---

**6. Avbrott och assemblerprogrammering med MC6809**.

**Lös *antingen* uppgift 6 *eller 7,* inte båda**
*Om du är registrerad på gamla kursen och haft MC6809-processorn löser du uppgift 6.*
*Om du är registrerad på nya kursen med MC12 löser du uppgift 7*

En pulsgenerator är ansluten via en avbrottsvippa till IRQ-ingången på en MC12-system.

Pulsgeneratorn har en frekvens på 100 Hz. För att nollställa avbrottsvippan krävs en läsning på den symboliska adressen IRQCLR. Pulsgeneratorn är den enda avbrottskällan anslutet till IRQ-ingången på processorn.

**a.** Du skall skriva en avbrottsrutin (IRQCNT) som läser en 8-bitars inport (IRQIN) och adderar det inlästa värdet till en 32-bitars variabel (IRQVAR). Både IRQIN och IRQVAR är variabler på tvåkomplementsform. **(6p)**

**b.** Skriv en initieringsrutin IRQINIT som initierar avbrottssystemet och som gör att IRQCNT anropas vid avbrott och att IRQVAR nollställs från början. **(4p)**

**7. Assemblerprogrammering för MC12**

**Lös *antingen* uppgift 6 *eller 7,* inte båda**
*Om du är registrerad på gamla kursen och haft MC6809-processorn löser du uppgift 6.*
*Om du är registrerad på nya kursen med MC12 löser du uppgift 7*

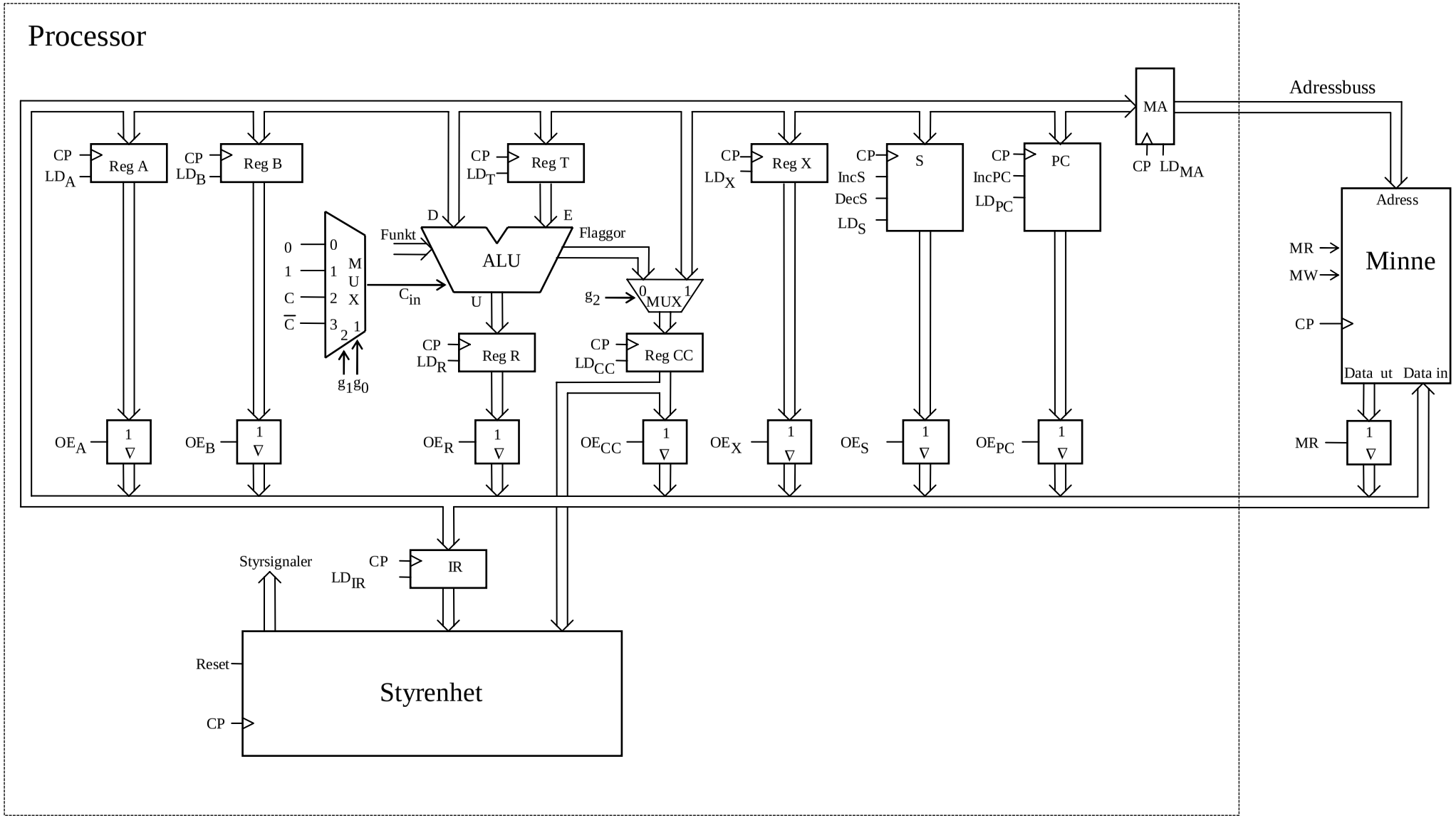
**a.** Skriv en programsekvens som nollställer $b_3$, ettställer bit $b_6$ och slutligen inverterar bitarna $b_7$ och $b_0$ i register B. **(3p)**

**b.** Skriv en subrutin **AntAA** som undersöker hur många gånger värdet $AA förekommer i en tabell i minnet. Tabellen innehåller 1000 bytes och är placerad med start på adress $3000 i minnet. Antalet upphittade $AA skall returneras i register D. Rita en flödesplan och skriv en väldokumenterad subrutin. **(6p)**

---

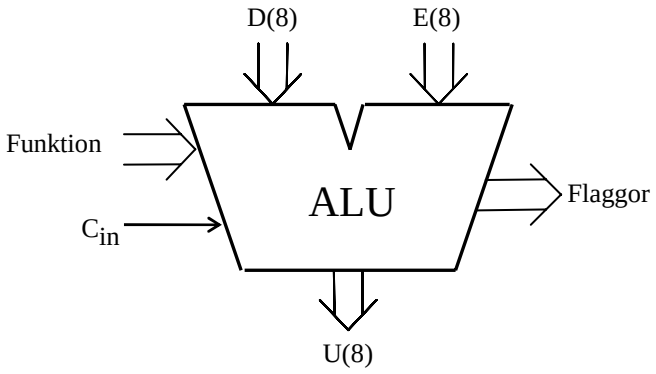
# FLEX-datorn

Bilaga 1



# ALU:ns funktion

Bilaga 2




ALU:ns **operation (logik- eller aritmetik-)** på indata **D**, **E** och $\mathbf{C_{in}}$ bestäms av insignalerna **Funktion [F = ($\mathbf{f_3}$, $\mathbf{f_2}$, $\mathbf{f_1}$, $\mathbf{f_0}$)]** enligt tabellen nedan**.**. I kolumnen Operation förklaras, när det behövs, hur operationen utförs. Med"**+**" och "–" avses **aritmetiska operationer**.

| $f_3$ $f_2$ $f_1$ $f_0$ | U = f(D,E,$C_{in}$) | |
|---|---|---|
| | **Operation** | **Resultat** |
| 0 0 0 0 | bitvis nollställning | 0 |
| 0 0 0 1 | | D |
| 0 0 1 0 | | E |
| 0 0 1 1 | bitvis invertering | $D_{1k}$ |
| 0 1 0 0 | bitvis invertering | $E_{1k}$ |
| 0 1 0 1 | bitvis OR | D OR E |
| 0 1 1 0 | bitvis AND | D AND E |
| 0 1 1 1 | bitvis XOR | D XOR E |
| 1 0 0 0 | D + 0 + $C_{in}$ | D + $C_{in}$ |
| 1 0 0 1 | D + FFH + $C_{in}$ | D – 1 + $C_{in}$ |
| 1 0 1 0 | | D + E + $C_{in}$ |
| 1 0 1 1 | D + D + $C_{in}$ | 2D + $C_{in}$ |
| 1 1 0 0 | D +$E_{1k}$ + $C_{in}$ | D – E – 1 + $C_{in}$ |
| 1 1 0 1 | | 0 |
| 1 1 1 0 | | 0 |
| 1 1 1 1 | bitvis ettställning | FFH |

Flaggorna är utsignaler och för de gäller:

**Carryflaggan (C)** är minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) när en aritmetisk operation utförs av ALU:n.

Vid **subtraktion** gäller för denna ALU att
**C = 1 om lånesiffra (borrow) uppstår och**
**C = 0 om lånesiffra inte uppstår**.

Carryflaggans värde är 0 vid andra operationer än aritmetiska.

**Overflowflaggan (V)** visar när en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik.

V-flaggans värde är 0 vid andra operationer än aritmetiska.

**Zeroflaggan (Z)** visar när en ALU-operation ger värdet noll som resultat på U-utgången.

**Signflaggan (N)** är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

**Half-carryflaggan (H)** är minnessiffran (carry) mellan de fyra minst signifikanta och de fyra mest signifikanta bitarna i ALU:n.

H-flaggans värde är 0 vid andra operationer än aritmetiska.

Tentamen i EDA451 Digital och Datorteknik, 2007-03-10

**1a)** R=X-Y utförs som R=X+$Y_{1komp}$+1 $Y_{1komp}$ = 01111.

| | |
|---|---|
| | 01111 |
| X | 00111 |
| +$Y_{1komp}$ | + 01111 |
| +1 | + 1 |
| =R | = 10111 |

**1b)** N=1; Z=0; V=1; $C_5$=0 $\Rightarrow$ C=1

**1c)** X=7; Y=16; R=23 (7-16=23, verkar rimligt ty C=1)

**1d)** X= 7; Y= -16: R=-9 (7-(-16)=-9, verkar rimligt ty V=1)

**1e)** $36_{16} = 110110_2$ på 6 bitar. Decimaltalet måste vara negativt.
$36_{2komp} = 001010_2 = 10_{10}$. Det sökta decimaltalet är –10

**1f)** 0001, 0001, 0011

**1g)** Udda paritet, ty antalet ettställda bitar är 7.

## Upg 2

**2a)** Uppgiften kan inte lösas då det krävs inverterare för att realisera en NAND-funktion med OR-grindar.

**2b)** Enligt tabellen till höger är HL≠VL.

| | | | VL | | | HL | | |
|---|---|---|---|---|---|---|---|---|
| **x** | **y** | **z** | **xy** | **y'z** | **f** | **y'** | **x'z** | **g** |
| 0 | 0 | 0 | 0 | 0 | **0** | 1 | 0 | **1** |
| 0 | 0 | 1 | 0 | 1 | **1** | 1 | 1 | **1** |
| 0 | 1 | 0 | 0 | 0 | **0** | 0 | 0 | **0** |
| 0 | 1 | 1 | 0 | 0 | **0** | 0 | 1 | **1** |
| 1 | 0 | 0 | 0 | 0 | **0** | 1 | 0 | **1** |
| 1 | 0 | 1 | 0 | 1 | **1** | 1 | 0 | **1** |
| 1 | 1 | 0 | 1 | 0 | **1** | 0 | 0 | **0** |
| 1 | 1 | 1 | 1 | 0 | **1** | 0 | 0 | **0** |

**2c)** Se figur 4.15 i kursboken.

**2d)** f(xyz) = (x+y+z)((xyz) + (xy + xz + yz)')

| **x** | **y** | **z** | **f** |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

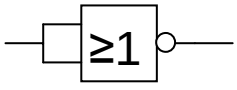
Minimering ger följande karnaughdiaram

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

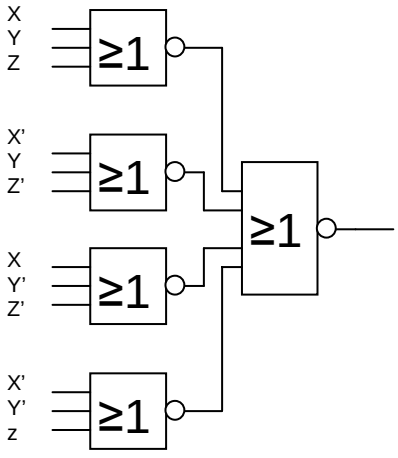
Ekvationen på konjunktiv form (NOR) blir:

f(xyz) = (x +y+z)(x'+y+z')(x+y'+z')(x'+y'+z)

Inverterade signaler skapas med en NOR-grind

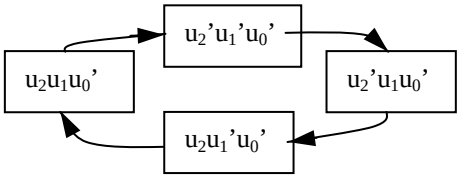



Lösningen ges till höger.



**3a)** Se sid 5.15, Figur 5.26 i kursboken

**3b)** **ASM-plan**: Hos en räknare används tillståndssignalerna q också som utsignaler u.



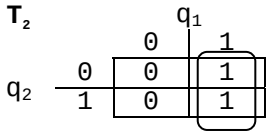

Studeras räknarens utsekvens ser vi att $q_0$ har värdet noll i alla tillstånd. Detta innebär att räknaren kan realiceras med endast två vippor.

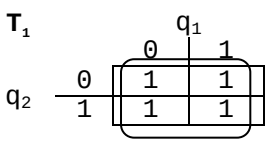
$q_0 = 0$. Sätter upp en tabell med $q_2q_1$.

| T | $Q^+$ |
|---|---|
| 0 | Q |
| 1 | Q' |

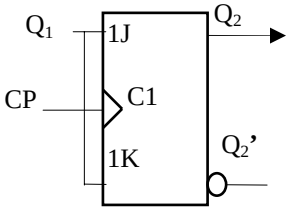
| Q | $Q^+$ | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

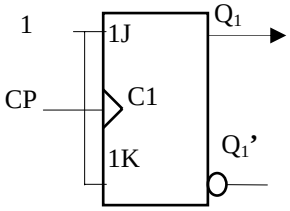
| Detta tillst $q_2q_1$ | Nästa tillst $q_2q_1$ | $T_2$ | $T_1$ |
|---|---|---|---|
| 00 | 01 | 0 | 1 |
| 01 | 10 | 1 | 1 |
| 10 | 11 | 0 | 1 |
| 11 | 00 | 1 | 1 |

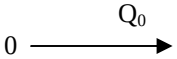



$T_2 = q_1$ $T_1 = 1$




## Upg 4

**4a)**

| State nr | RTN-beskrivning | Styrsignaler (=1) |
|---|---|---|
| 0 | PC→MA, PC+1→PC | $OE_{PC}$, $LD_{MA}$, IncPC |
| 1 | M→MA | MR, $LD_{MA}$ |
| 2 | M→T | MR, $LD_T$ |
| 3 | A+T→R, Flaggor→CC | $OE_A$, $LD_R$ , $f_3f_1$, $LD_{CC}$ |
| 4 | R→A, NF | $OE_R$, $LD_A$, NF |

**4b)**

0) Förbered för läsning av adressoperand i minnet, Öka PC med ett
1) Läs adressoperanden från minnet till register MA för att förbereda läsning av data
2) Läs data från minnet till register T
3) Addera data och register A. Spara summan i register R och påverka flaggbitarna.
4) Flytta summan till register A

Instruktionen är ADDA Adr

**4c)**

| State nr | RTN-beskrivning | Styrsignaler (=1) |
|---|---|---|
| 0 | S-1→S | DecS |
| 1 | S→MA | $OE_S$, $LD_{MA}$ |
| 2 | PC→M | $OE_{PC}$, MW |
| 3 | X→PC | $OE_X$, $LD_{PC}$, NF |

## Upg 5

**5a)** DEC Adr utför M(Adr)-1→ M(Adr)

```
        PSHA
        LDA    Adr
        ADDA   #$FF          DECA = SUBA #1 = ADDA #$FF
        STA  Adr
        PULA
```

**5b)** BEQ-instruktionen upptar två bytes i minnet, vilket innebär att ”från-adressen” är $C9+2 = $CB. ”Till-adressen” är $A6. Då beräknad offset här blir $DB blir maskininstruktionens offset $DB.
Maskininstruktionen blir $5D $DB.

| | ~~1~~ ~~10~~…10 |
|---|---|
| Till Adr | $ A 6 |
| - Från Adr | - $ C B |
| = Offset | = $ D B |

**6a)**

```
ANDB    #%11110111    Nollställ
ORAB    #%01000000    Ettställ
EORB    #%10000001    Invertera
```

**6b)**

```
* Subrutinen AntAA letar upp antalet förekomster
* av värdet $AA i en tabell med start på adress
* $3000. Tabellen innehåller 1000 element.
*
* Indata: -
* Utdata:  Antalet $AA i register D
* Påverkade register: -
* Anrop: jsr  AntAA
* Programmerare: Rolf 070329

AntAA   pshx
        pshy
        pshc
        ldx    #$3000       Tabellstart
        ldy    0            Antal = 0
        ldaa   #$AA         Sökmönster
mera    cmpa   ,x+          Jämför
        bne    ejAA         o hoppa om olika
        iny                 annars öka antal
ejAA    cmpx   #$3000+1000  Sista?
        bne    mera         hoppa om ej sista
        pulc
        puly
        pulx
        rts
```